Kyoto Constella Technologies Co., Ltd

CzeekS Manual

December 4, 2014

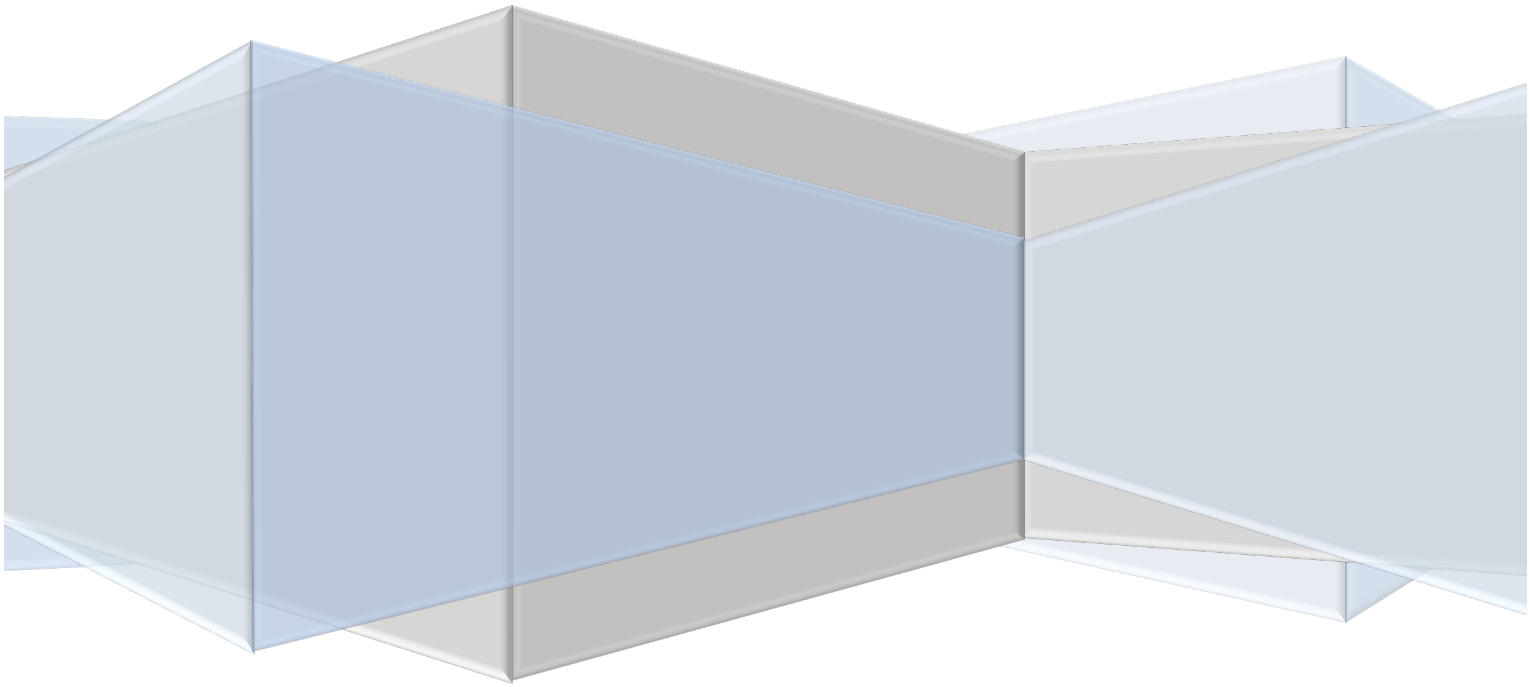

TABLE OF CONTENTS

1. Introduction 1

2. Installation and Settings 2

2-1. Extracting Archive Files and Placement of License File 2

2-2. Setting Environmental Variables 2

2-3. OpenBabel Settings 3

3. Compound Screening and Target Prediction 3

3-1. CGBVS Model 3

3-2. Compound Screening (from descriptor calculation to scoring) 5

3-3. Target Prediction 8

3-4. Calculation of Structure Similarity (Tanimoto Coefficient) 9

4. Creation of CGBVS Model and Addition of User Data 10

4-1. Data and Format Required for Model Creation 10

4-2. Creation of Model File (DB File) 11

4-3. Addition of Data 12

4-4. Machine Learning 12

4-5. Others 12

5. cgbvs Command Reference 14

Trademarks

All the company and product names appearing in this manual are trademarks or registered trademarks of the respective companies. Furthermore, trademarks are not appended to all the software and product names described in this manual.

©2012 Kyoto Constella Technologies Co., Ltd
All Rights Reserved. Copyright 2014

1. Introduction

In recent years, it has become common sense to have view that a certain compound can interact with multiple target proteins. We refer to such complicated compound-protein relationship as chemical genomics information. It is this kind of information that has been built into a bioactivity database and continuously improved by organizations such as ChEMBL. We refer to the technique of predicting and screening the activity of an unknown compound by pattern recognition of such information through machine learning as CGBVS (Chemical Genomics-Based Virtual Screening). CzeekS is a set of tools for performing CGBVS and offers the following functions.

- Compound scoring
- Creation of CGBVS learning models
- Managing functions of learning models
- Calculation of compound fingerprints (MACCS)
- Similarity calculation with a target compound

Section 2 of this manual explains the installation method of CzeekS. Section 3 explains the screening method of a compound using sample data. Selectivity and target prediction of a compound as advanced utilities are also explained in the same Section. Section 4 explains the construction of a learning model using sample data. Section 5 describes command references.

Using CzeekS in the following computer environment is recommended. Since CzeekS supports the parallel computation by OpenMP, more CPU cores equates to better efficiency. It is also possible to run CzeekS using two or more machines.

CPU	Multi core CPU with four or more cores (Intel, AMD)
Memory	8GB or more
HDD	10 GB or more of free space
OS	CentOS5.x or 6.x 64bit (Linux kernel 2.6)
External tool	DRAGON ver. 6.0.30
External library	OpenBabel 2.3.1

Time required for machine learning of sample data (1 node)

CPU	Number of threads	Memory	Computation time
Intel Xeon E5620 × 2	16	24GB	20h 10m
Intel Core i3 550	4	4GB	66h 52m
AMD Phenom Ⅱ X6 1055T	6	8GB	70h 40m

2. Installation and Settings

2-1. Extracting Archive Files and Placement of License File

Extract the archive file "CzeekS_******.tgz" using the tar command as follows. While you can extract into any one of directories, it is recommended to extract it under /usr/local or under /home/czeeks after creating users such as czeeks. In this manual we proceed with the explanations with the assumption that files were extracted under /home/czeeks.

```
$ tar xvfz CzeekS_******.tgz⏎
CGBVS/
CGBVS/exec/
CGBVS/exec/license.dat
CGBVS/exec/cgbvs
CGBVS/exec/calc_dragon.sh
CGBVS/exec/2D_990.drt
CGBVS/exec/calc_FP_MACCS
CGBVS/exec/SVMlearn
CGBVS/exec/protein.lst
・・・
```

Extracted files are indicated below. Copy your license file (license.dat file received from Constella) into the subdirectory /home/czeeks/CGBVS/exec overwriting the existing invalid license.dat file.

```
CGBVS
|-- example                 Directory in which sample data and other files.were extracted
|   |-- gpcr.csv            Descriptor vector of GPCR
|   |-- positive.csv        Positive examples
|   |-- sample_mols.csv     Descriptor file of test compounds
|   |-- sample_mols.fp      Fingerprint file of test compounds
|   |-- sample_mols.sdf     SD file of test compounds
|   |-- sample_mols.smi     SMILES file of test compounds
|   |-- training_mols.csv   Descriptor file of sample compounds for learning
|   |-- training_mols.fp    Fingerprint file of sample compounds for learning
|   |-- training_mols.sdf   SD file of sample compounds for learning
|   `-- training_mols.smi   SMILES file of sample compounds for learning
`-- exec                    Directory in which executable files. were extracted
    |-- 2D_894.drt          Script file for DRAGON6
    |-- SVMlearn            SVM machine learning executable file
    |-- calc_FP_MACCS       MACCS fingerprints calculation executable file
    |-- calc_dragon.sh      DRAGON6 script for descriptor calculation
    |-- cgbvs               CGBVS executable file
    |-- license.dat         License file (invalid initially)
    `-- protein.lst         Protein list file
```

2-2. Setting Environmental Variables

After extracting the files and copying your license file, set environment variables as indicated below. Add the same details into the .bashrc file.

```
$ export CGBVS=/home/czeeks/CGBVS/exec⏎
$ export PATH=$PATH:$CGBVS⏎
$ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH⏎
$ export DRAGON6=/usr/local/bin⏎  The path in which DRAGON6 was installed
```

For the environment variables of DRAGON6, please specify the directory where the DRAGON6 executable file "dragon6shell" is installed. Also specify file name with a full path in environmental-variable CGBVS_LICENSE if you want to put the license file license.dat in a subdirectory other than under ${CGBVS}.

2-3. OpenBabel Settings

Within CzeekS, OpenBabel is used for the calculation (using calc_FP_MACCS) of compound fingerprints (MACCS) and generation of SMILES from SD file. If OpenBabel is not yet installed in your system, you can install it using the following steps.

① Installation of cmake

Since cmake is required to compile OpenBabel, it has to be installed into the system. It can be installed using the command "yum install cmake" after becoming a superuser.

② Compiling and Installing OpenBabel

OpenBabel is a free software (GPL v.2) and can be downloaded from the following URL.

http://openbabel.org/wiki/Get_Open_Babel

Extract the archive file after downloading it from the URL above. If the version you downloaded is 2.3.1 and the archive file is extracted using the tar command, a directory named openbabel-2.3.1 will be created containing the extracted file(s). Switch into the openbabel-2.3.1 directory then compile and install OpenBabel using the following steps.

```
$ mkdir build⏎        Create a suitable directory.
$ cd build⏎
$ cmake ../⏎          Execute the cmake command.
$ make⏎               Compile OpenBabel.
$ su⏎                 Become a superuser.
# make install⏎       Install it in the default path.
```

The above procedure is for the necessary minimum installation of OpenBabel for use within CzeekS. Refer to the OpenBabel manual or other sources for detailed compile settings.

3. Compound Screening and Target Prediction

3-1. CGBVS Model

Sample model files are included in CzeekS and these should not be used for actual *in silico* screening. The extension of a model file is ".db", and hereinafter may be referred to as "DB file." These samples models are created from data originating from the ChEMBL database. Those data are also included in CzeekS. Section 4 gives an explanation about these data..

In CGBVS, the support vector machine (SVM) is used as the pattern recognition technique. SVM is the method of classifying two classes of positive examples and negative examples, and both data are required to perform

machine learning. However, while there are plenty of information about interacting compound-protein pairs (positive examples), there are very few information about experimentally validated non-interacting compound-protein pairs (negative examples) available in public databases. In this case, information to be used as negative examples is generated virtually before performing machine learning. Virtual negative examples are generated by rearranging positive example pairs at random. This creates multiple sets of negative examples that are used to create learning models. The average scores of negative example sets are then calculated and eventually used.

Scores generated by CGBVS are of two types. One is the average of the decision function value of SVM and it takes the range of -∞ - +∞. Another is the average of this decision function value after normalization by sigmoid function and takes the range 0-1. Usually, the normalized score is displayed in CzeekS. This score indicates the probability of the compound having an activity against the target protein. This does not indicate proportionality between this value and the value indicating actual activity.

The information on the CGBVS model explained above can be checked by the "cgbvs status" command. Check the DB file of sample models first by using the following command. The information about the number of the compounds registered in the DB file, the number of the proteins, and the learned models are displayed in the list.

```
$ cgbvs status gpcr_sample.db⏎
[compound]
Dragon6 v.6.0.30              Software used to generate the compound descriptors
# of data = 13838             Number of compounds registered
# of descriptors = 894        Number of compound descriptors

[protein]
PROFEAT 2011                  System used to generate the protein descriptors
# of data = 859               Number of the proteins registered
# of descriptors = 1080       Number of protein descriptors

[fingerprint]
MACCS                         Type of fingerprints
# of data = 13838             Number of the compounds registered

[interactions]
# of positive interactions = 21761   Interaction information on the positive example
# of negative interactions = 0       Interaction information on the negative example

[details of models]
# of sampled positive interactions = 21761  The number of interactions used for machine
                                            learning
|  id  |   nSV   |  dim  |    C    |  gamma  | accuracy |
|------+---------+-------+---------+---------+----------|
|    1 |   41024 |  1974 | 10.0000 |  0.0100 |  82.2664 |
|    2 |   41026 |  1974 | 10.0000 |  0.0100 |  82.2019 |
|    3 |   41007 |  1974 | 10.0000 |  0.0100 |  82.3506 |
|    4 |   41023 |  1974 | 10.0000 |  0.0100 |  82.0856 |
|    5 |   41046 |  1974 | 10.0000 |  0.0100 |  82.1124 |
```

Concerning the table “details of models”, “id” indicates the ID number of the model and, in this case, 5 are shown. “nSV” indicates the number of support vectors while “C” and “gamma” indicate parameters for SVM.

“Accuracy” indicates the precision of distinction when cross-validation is performed for each model. The table of the proteins that are available for calculation will be displayed if the -p option is used with the "cgbvs status" command.

```
$ cgbvs status -p gpcr_sample.db⏎
[protein ID list]
protein ID      # of compounds  accession      name
5HT1A_HUMAN           407      P08908  5-hydroxytryptamine receptor 1A
5HT1B_HUMAN           207      P28222  5-hydroxytryptamine receptor 1B
5HT1D_HUMAN           203      P28221  5-hydroxytryptamine receptor 1D
5HT1E_HUMAN            74      P28566  5-hydroxytryptamine receptor 1E
5HT1F_HUMAN           103      P30939  5-hydroxytryptamine receptor 1F
5HT2A_HUMAN           388      P28223  5-hydroxytryptamine receptor 2A
5HT2B_HUMAN           287      P41595  5-hydroxytryptamine receptor 2B
5HT2C_HUMAN           422      P28335  5-hydroxytryptamine receptor 2C
5HT4R_HUMAN           109      Q13639  5-hydroxytryptamine receptor 4
5HT5A_HUMAN           112      P47898  5-hydroxytryptamine receptor 5A
5HT6R_HUMAN           252      P50406  5-hydroxytryptamine receptor 6
5HT7R_HUMAN           227      P34969  5-hydroxytryptamine receptor 7
A4_HUMAN              100      P05067  Amyloid beta A4 protein
```

The “protein ID” shown in the table indicates the protein ID used during binding prediction calculation. This ID, including the “accession” are the same IDs being used in the protein database UniProt (http://www.uniprot.org). The “# of compounds” column indicates the number of active compounds for every protein registered in the DB file. While it depends on the diversity of the compound structure, there is a general trend that higher number of compounds results to more accurate prediction calculation.

3-2. Compound Screening (from descriptor calculation to scoring)

Descriptor Calculation

It is necessary to calculate the descriptors from compound structures (SD file) before compound prediction calculation against target protein(s) can be performed. The type of the compound descriptor must coincide with the type in the DB file. Furthermore, it is also necessary to make the compound processing conditions (desalting, charge neutralization, etc.) uniform at the time of descriptor calculation. The descriptor of the file included in CzeekS as a sample has been obtained through calculation by DRAGON6 using the script file under directory exec, and the compounds are desalted and the charges are neutralized.

Calculation of descriptors from SMILES file using DRAGON6 can be performed using the command below. This command creates a standard output file. You can use OpenBabel to convert SD files to SMILES files.

```
$ babel -isdf sample_mols.sdf -osmi sample_mols.smi⏎
$ calc_dragon.sh sample_mols.smi > output.csv⏎
$ cat output.csv⏎
ZINC00074638,315.320,8.522,24.952,38.109,25.091,…
ZINC00075927,269.300,8.416,21.796,32.563,22.216,…
ZINC00492910,300.390,7.152,25.928,42.138,27.228,…
ZINC02759964,339.170,10.941,21.362,32.153,21.784,…
ZINC03518134,264.360,6.778,22.928,39.138,24.228,…
……
```

Execute when there is no SMILES file.

Format will be comma separated values (CSV).

Descriptor file should show information of only 1 compound per line, with the following information written in a comma-delimited manner: Compound ID, Descriptor1, Descriptor2, etc. Be careful of the format, especially when not using the calc_dragon.sh script.

Scoring

Prediction calculation can be performed using the "cgbvs predict" command once the descriptor file has been prepared. The sample descriptor file (sample_mols.csv) included in the CzeekS installation is the same file created using the command above. For example, the score calculation against adrenaline β2 receptor can be performed using the following command and the result is subsequently displayed on the screen.

```
$ cgbvs predict gpcr_sample.db ADRB2_HUMAN sample_mols.csv⏎
compound        ADRB2_HUMAN
ZINC00074638     0.28596379
ZINC00075927     0.20458141
ZINC00492910     0.94327482
ZINC02759964     0.20639719
ZINC03518134     0.23033582
ZINC03912658     0.20744996
ZINC04143221     0.20678472
……
```

Argument 2 of this command specifies the DB file of the CGBVS model. Argument 3 specifies the target protein ID and the file name of the compound descriptor is specified by argument 4. Please check the available target proteins that can be specified in argument 3 above by using the "cgbvs status -p" command. You can redirect the calculation results to a file if needed.

Scoring against multiple proteins

Scoring against multiple proteins can be performed by specifying 2 or more target proteins separated by commas in argument 3. There is no limit to the number of target proteins that can be specified. For example, execute the following command if you want to calculate scores against β1 and β2 receptors.

```
$ cgbvs predict gpcr_sample.db ADRB1_HUMAN,ADRB2_HUMAN sample_mols.csv⏎
compound        ADRB1_HUMAN     ADRB2_HUMAN
ZINC00074638    0.17813841      0.28596379
ZINC00075927    0.20430067      0.20458141
ZINC00492910    0.95634899      0.94327482
ZINC02759964    0.20634203      0.20639719
ZINC03518134    0.20936986      0.23033582
ZINC03912658    0.20745000      0.20744996
ZINC04143221    0.20458645      0.20678472
......
```

The scores are then displayed in a tab-delimited manner. If multiple proteins are specified, screening with consideration to compound selectivity. The % sign can be used as a wild card. For example, screening against all the adrenalin receptors, including α receptors, can be performed using the following command.

```
$ cgbvs predict gpcr_sample.db ADA%,ADR% sample_mols.csv⏎
compound        ADA1A_HUMAN     ADA1B_HUMAN     ADA1D_HUMAN     ADA2A_HUMAN
ADA2B_HUMAN     ADA2C_HUMAN     ADRB1_HUMAN     ADRB2_HUMAN     ADRB3_HUMAN
ZINC00074638    0.12149832      0.12341347      0.13156714      0.17294950
0.17890952      0.16551650      0.17813841      0.28596379      0.15600113
ZINC00075927    0.20223752      0.20377914      0.19969655      0.20499859
0.20498811      0.20679086      0.20430067      0.20458141      0.20357125
ZINC00492910    0.66670499      0.58061474      0.46289849      0.12777100
0.17438357      0.29246626      0.95634899      0.94327482      0.93282221
......
```

Display format

The display information of the CGBVS score can be changed through the “cgbvs predict” command option. The average of the decision function score of SVM instead of the normalized score can be displayed when the –d option is used.

```
$ cgbvs predict -d gpcr_sample.db ADR% sample_mols.csv⏎
compound        ADRB1_HUMAN     ADRB2_HUMAN     ADRB3_HUMAN
ZINC00074638    -0.26372672     -0.19973609     -0.28057862
ZINC00075927    -0.24563104     -0.24544712     -0.24609816
ZINC00492910    0.22043506      0.19194762      0.17238724
ZINC02759964    -0.24432048     -0.24428153     -0.24424564
ZINC03518134    -0.24301969     -0.23186666     -0.25030520
ZINC03912658    -0.24361160     -0.24361162     -0.24361155
ZINC04143221    -0.24544375     -0.24403326     -0.24620981
......
```

Both the decision function value and the normalized score are displayed when using the -v option.

```
$ cgbvs predict -v gpcr_sample.db ADR% sample_mols.csv⏎
compound        protein probability     score
ZINC00074638    ADRB1_HUMAN     0.17813841      -0.26372672
ZINC00074638    ADRB2_HUMAN     0.28596379      -0.19973609
ZINC00074638    ADRB3_HUMAN     0.15600113      -0.28057862
ZINC00075927    ADRB1_HUMAN     0.20430067      -0.24563104
ZINC00075927    ADRB2_HUMAN     0.20458141      -0.24544712
ZINC00075927    ADRB3_HUMAN     0.20357125      -0.24609816
ZINC00492910    ADRB1_HUMAN     0.95634899      0.22043506
ZINC00492910    ADRB2_HUMAN     0.94327482      0.19194762
ZINC00492910    ADRB3_HUMAN     0.93282221      0.17238724
……
```

In this format, 2 types of scores for a compound-protein pair are displayed in one line.

3-3. Target Prediction

Target Prediction Using CGBVS

The preceding section explained that using CGBVS enables scoring against multiple proteins. Extending this view, if score is calculated against all available proteins, it makes the search for the target protein possible.

When specifying the target argument of “cgbvs predict” and the “all” option is used, all the compounds registered in the DB file will be scored against all proteins available. Also use the –a option if you want to score against proteins that do not have registered ligands in the DB file.

(Available proteins can be checked by cgbvs status –pv command) For example, calculating scores for the compound with the ID ZINC10454282 in the “sample_mols.csv” file against all the proteins available can be performed as follows:

```
$ grep ZINC10454282 sample_mols.csv > test.csv⏎
$ cgbvs predict -v gpcr_sample.db all test.csv⏎
compound        protein probability     score
ZINC10454282    5HT1A_HUMAN     0.20230991      -0.25578423
ZINC10454282    5HT1B_HUMAN     0.21639315      -0.24077885
ZINC10454282    5HT1D_HUMAN     0.23220133      -0.22949139
ZINC10454282    5HT1E_HUMAN     0.55664237      -0.07965085
ZINC10454282    5HT1F_HUMAN     0.25697899      -0.21507697
ZINC10454282    5HT2A_HUMAN     0.26910340      -0.21419708
ZINC10454282    5HT2B_HUMAN     0.33050923      -0.17881329
ZINC10454282    5HT2C_HUMAN     0.22229833      -0.23952181
ZINC10454282    5HT4R_HUMAN     0.20269564      -0.24918873
ZINC10454282    5HT5A_HUMAN     0.38818196      -0.15109197
ZINC10454282    5HT6R_HUMAN     0.25142398      -0.21765020
ZINC10454282    5HT7R_HUMAN     0.20856294      -0.24860986
ZINC10454282    A4_HUMAN        0.19385333      -0.25367056
ZINC10454282    AA1R_HUMAN      0.13968021      -0.29825117
……
```

In this example, the –v option is used to display the protein ID in a column. Sorting the probability scores from highest to lowest can be done by redirecting the output to a file, and then having it sorted by using the commands below.

```
$ cgbvs predict -v gpcr_sample.db all test.csv > out⏎
$ sort -k3 -nr out | head⏎
ZINC10454282    MTR1A_HUMAN     0.86593198      0.09740989
ZINC10454282    MTR1B_HUMAN     0.82153994      0.05707536
ZINC10454282    TSHR_HUMAN      0.71460631      -0.00721098
ZINC10454282    GRM2_HUMAN      0.71075249      -0.00930970
ZINC10454282    5HT1E_HUMAN     0.55664237      -0.07965085
ZINC10454282    CCR3_HUMAN      0.50475269      -0.10143637
ZINC10454282    ACM3_HUMAN      0.43933527      -0.12913799
ZINC10454282    ACM5_HUMAN      0.42168349      -0.13881759
ZINC10454282    HRH3_HUMAN      0.40058001      -0.14602600
ZINC10454282    ACM4_HUMAN      0.39069602      -0.15187816
```

Information about the two proteins on top of the column, MTR1A_HUMAN and MTR1B_HUMAN can be displayed by issuing the command below.

```
$ cgbvs status -pv gpcr_sample.db | grep -e "MTR1..*"⏎
MTR1A_HUMAN           102     P48039  Melatonin receptor type 1A
MTR1B_HUMAN           101     P49286  Melatonin receptor type 1B
```

3-4. Calculation of Structure Similarity (Tanimoto Coefficient)

With CzeekS, the Tanimoto coefficient (Similarity) can be calculated from the fingerprints of the compound. Tanimoto coefficient is calculated based on the specified target protein and the information of compounds (in DB file) to be evaluated. The Tanimoto coefficient of multiple compounds is calculated and the maximum value is displayed. This is performed by issuing the “cgbvs predict -s” command. The procedure is shown below.

```
$ calc_FP_MACCS sample_mols.sdf test.fp⏎     Fingerprints calculation. test.fp and
                                              sample_mols.fp will be the same.
$ cgbvs predict -s gpcr_sample.db ADRB2_HUMAN test.fp
compound        ADRB2_HUMAN
ZINC00074638    0.55737705
ZINC00075927    0.48571429
ZINC00492910    0.71428571
ZINC02759964    0.58108108
ZINC03518134    0.56666667
ZINC03912658    0.72000000
ZINC04143221    0.72972973
ZINC05766699    0.54385965
```

The contents of the fingerprint file test.fp are shown below..

```
$ head sample_mols.fp⏎
ZINC00074638,42 50 57 62 72 75 76 83 85 87 89 91 92 95…
ZINC00075927,41 42 52 65 75 78 80 87 92 94 95 97 98 107 110…
ZINC00492910,54 72 82 90 92 95 97 100 104 109 110 113 117 126…
ZINC02759964,24 46 49 52 56 63 65 70 71 75 79 80 83 87 92 93…
ZINC03518134,65 72 75 83 85 90 91 92 93 95 96 104 110 111 117…
…
```

Regarding the format, the first column shows the compound ID while the next column shows the fingerprints.

The numbers in the fingerprint part are generally increasing values (from left to right) corresponding to the positions of "1" within a list of binary values (bitstrings) created during evaluation of compound structures based on MACCS keys.

4. Creation of CGBVS Model and Addition of User Data

4-1. Data and Format Required for Model Creation

The following are required for the creation of a CGBVS learning model

1. Compound descriptor information
2. Protein descriptor information
3. Compound-protein pair interaction information

The above-mentioned information must be prepared as comma-delimited (CSV) files. The file format is described as follows using the sample data for model creation as an example. The contents of the sample file "training_mols.csv" are shown below.

```
$ head training_mols.csv⏎
1000029,419.62,6.557,38.396,63.214,41.347,72.142,0.6,0.988,0.646,…
1000123,279.35,8.73,21.03,32.782,21.835,36.119,0.657,1.024,0.682,…
100014,377.35,8.029,30.009,46.891,32.353,53.033,0.638,0.998,0.688,…
1000194,405.5,7.651,33.993,53.443,35.245,59.857,0.641,1.008,0.665,…
1000948,246.24,8.794,19.009,29.047,18.875,31.495,0.679,1.037,0.674,…
1000956,399.54,9.08,30.072,44.618,31.801,49.242,0.683,1.014,0.723,…
1001098,216.32,6.76,19.246,31.709,20.591,36.484,0.601,0.991,0.643,…
1001421,300.51,8.839,22.007,33.945,24.739,37.872,0.647,0.998,0.728,…
100163,481.66,6.784,42.746,70.829,45.466,80.149,0.602,0.998,0.64,…
1001651,336.37,8.204,27.59,41.698,28.159,45.741,0.673,1.017,0.687,…
```

It is the same format as the descriptor file in Section 3 used for the scoring of compounds. The first column shows the compound ID while the numerical values are indicated starting at column 2. This is the result of calculating the descriptors from the SMILES file "training _mols.smi" using DRAGON6.

Regarding protein descriptors, the format is essentially the same as that for compounds. A sample file (gpcr.csv) is shown below.

```
$ head gpcr.csv⏎
5HT1A_HUMAN,9.71564,3.317536,3.791469,3.554502,4.028436,…
5HT1B_HUMAN,8.974359,2.820513,3.589744,3.333333,4.358974,…
5HT1D_HUMAN,9.814324,2.917772,2.65252,3.183024,4.509284,…
5HT1E_HUMAN,6.575342,3.287671,3.561644,3.287671,4.657534,…
5HT1F_HUMAN,6.284153,3.005464,4.098361,4.644809,4.371585,…
5HT2A_HUMAN,6.157113,3.184713,4.246285,3.821656,5.307856,…
5HT2B_HUMAN,6.029106,1.663202,2.910603,4.365904,5.405405,…
5HT2C_HUMAN,5.895197,2.620087,2.838428,4.803493,4.585153,…
5HT4R_HUMAN,6.958763,4.639175,3.865979,3.092784,5.670103,…
5HT5A_HUMAN,7.843137,2.80112,2.521008,3.921569,6.162465,…
```

The example above is calculated from FASTA file using the PROFEAT site (the link is indicated below)..

http://bidd.cz3.nus.edu.sg/cgi-bin/prof/protein/profnew.cgi

Refer to the PROFEAT site for detailed information including the calculation method and other relevant information. CzeekS adopts the UniProt ID as the protein ID, and as much as possible, if the protein is not considered to be a special protein, , please use the "*_HUMAN" format.

Regarding the interaction information, the contents of the sample file "positive.csv" by the command shown below.

```
$ head positive.csv⏎
1000029,NPBW1_HUMAN
1000123,ARBK1_HUMAN
100014,CRFR1_HUMAN
1000194,FAK2_HUMAN
1000948,CCR6_HUMAN
1000956,NTR1_HUMAN
1001098,FAK2_HUMAN
1001421,OX1R_HUMAN
100163,PTAFR_HUMAN
1001651,ADRB2_HUMAN
```

In the format above, the compound ID is shown in the first column while the protein ID is in the second column. In this way, a compound-protein pair is shown in one line. In this example, we utilized data from the ChEMBL database where only compound-protein combinations having activities of 30μM or less are selected.

4-2. Creation of Model File (DB File)

The CGBVS model file (DB file) can be created once the required files above are prepared. Here, we will be using the sample files (training_mols.csv, gpcr.csv, positive.csv) introduced earlier. Perform the operation by issuing the following commands.

```
$ cgbvs create training.db⏎ Creation of an empty DB file
$ cgbvs import training.db training_mols.csv compound⏎ Registration of compound descriptors
import training_mols.csv
$ cgbvs import training.db gpcr.csv protein⏎ Registration of protein descriptors
import gpcr.csv
$ cgbvs import training.db positive.csv positive⏎ Registration of interaction information
import positive.csv
```

First, an empty DB file is created. Next, the 3 required files are imported into the DB file (files can be imported in any order). File import and DB file creation can be done simultaneously by using the appropriate option with the “cgbvs create” command. At this point, the CGBVS model can be created by performing machine learning. Please refer to section 4-4 for details about machine learning.

As explained in section 3-4, calculation of structure similarity (Tanimoto coefficient) of the compounds registered in the DB file can be performed in CzeekS. When calculating structure similarity, compound descriptors and fingerprints must be registered first. Fingerprint registration uses the following command.

```
$ cgbvs import training.db training_mols.fp fingerprint⏎
import training_mols.fp
```

Refer to section 3-4 for the format of the fingerprint file and the calculation method using MACCS.

4-3. Addition of Data

This section describes how to update the CGBVS model by adding data (user's original assay data) separately to the existing DB file. There are basically three types of information that must be prepared as described in section 4-1. However, it is not anymore necessary to prepare the protein descriptor information. To check whether the intended target protein is registered or not, execute the "cgbvs status" with the –pv option. The –pv option will also display proteins with 0 ligand. Please refer to section 3-1 for more information.

Use the "cgbvs add" command in order to add data to the DB file. As sample data, 100 ligands of the histamine H3 receptor are prepared as a file called H3_mols.sdf. The calculated descriptors for these ligands are contained in the file H3_mols.csv. The interaction information file is H3_positive.csv. As the protein descriptor is already registered, there is no necessity for any addition.

```
$ cgbvs add training.db H3_mols.csv compound⏎   Addition of compound descriptors
import H3_mols.csv
$ cgbvs add training.db H3_positive.csv positive⏎   Addition of interaction information
import H3_positive.csv
```

4-4. Machine Learning

After registering or adding data to the DB file, it is necessary to perform machine learning using SVM. Machine learning can be executed as follows using the "cgbvs learn" command.

```
$ cgbvs learn -c 10 -g 0.01 training.db 5⏎
output input_1
SVMlearn -c 10.000000 -g 0.010000 -v 5 input_1 model_1⏎
    itr       nSV      vKKT                Objective
      1       978     42378    -4.497671328644441E+02
      2      1907     41404    -8.200883693534472E+02
      3      2786     43240    -1.321260914509097E+03
```

The above-mentioned example will create five sets of negative examples and this is specified in the last argument. 5-10 is usually specified for this argument. Refer to section 3-1 for details about the negative example set. -c and -g are the optional parameters of SVM. The parameter C relating to the soft margin of SVM is specified by -c. In CzeekS, the gauss type RBF (Radial Basis Function) function is employed as the kernel function of SVM. The value γ of the RBF function is specified by -g.

Although machine learning is executed assuming C=10 and γ=0.01 in the above example, predictive accuracy depends on the SVM parameter value. It is recommended to check different combinations of C and γ in order to find the optimal settings. An example of parameter search is described in the next section.

4-5. Others

In 4-4, the machine learning execution method was described where calculation was performed by creating 5 sets of negative examples. When utilizing several machines, it is also possible to calculate in parallel for these negative

example sets. Here, command execution, is described regarding how to perform machine-learning calculation independently (in parallel) for every negative example set. First, create the SVM input files by using the –f option with the “cgbvs learn” command as indicated below.

```
$ cgbvs learn –f training.db 5⏎
output input_1
output input_2
output input_3
output input_4
output input_5
```

Next, execute SVM machine learning for each machine as follows.

```
$ SVMlearn -c 10 -g 0.01 input_1 model_1⏎   Execute for machine 1
$ SVMlearn -c 10 -g 0.01 input_2 model_2⏎   Execute for machine 2
$ SVMlearn -c 10 -g 0.01 input_3 model_3⏎   Execute for machine 3
$ SVMlearn -c 10 -g 0.01 input_4 model_4⏎   Execute for machine 4
$ SVMlearn -c 10 -g 0.01 input_5 model_5⏎   Execute for machine 5
```

If the above-mentioned command has successfully completed, five files named model_1 to model_5 should already exist. Import those into the DB file by using the following commands.

```
$ cgbvs add_model training.db model_1 1⏎   Import model_1 as id=1
$ cgbvs add_model training.db model_2 2⏎   Import model_2 as id=2
$ cgbvs add_model training.db model_3 3⏎   Import model_3 as id=3
$ cgbvs add_model training.db model_4 4⏎   Import model_4 as id=4
$ cgbvs add_model training.db model_5 5⏎   Import model_5 as id=5
```

Imported models can be checked using the “cgbvs status” command. Searching for the optimal SVM parameters can also be performed using the above method. The following is an example script that searches for optimal parameters of the file “input_1”.

```
#!/bin/sh

for c in 1 3 10 30 100; do
 for g in 0.001 0.003 0.01 0.03 0.1; do
  echo -ne $c"¥t"$g"¥t"
  SVMlearn -c $c -g $g input_1 model_1 | grep cross-validation | awk '{print $6}'
 done
done
```

The above script will calculate for SVM parameters using a total of 25 combinations of γ (0.001, 0.003, 0.01, 0.03, 0.1) and C (1, 3, 10, 30, 100) values. Output is displayed in the order of C, γ, and prediction rate. Calculate for the combination of C and γ that will give the highest prediction rate for each model then import the results into the DB file.

5. cgbvs Command Reference

Usage

cgbvs <subcommand> [<option>] <Argument>

The available subcommands are as follows: add, add_model, comment, create, delete, del_model, import, learn, predict, status. Note that <option> and <Argument> may differ for every subcommand.

Subcommands

add: Used to append data into the DB file

(Format)

cgbvs add <db file> <data file> <target>

(Description)

Use the "add" subcommand to append data files (CSV), such as descriptor information and interaction pair information to existing data in the DB file. Also specify the type of the data files (descriptor information, interaction pair information, etc. of the compound) in the <target> argument. The types of the targets that can be specified are as follows.

compound	Compound descriptors
protein	Protein descriptors
positive	Positive interaction pairs (positive examples)
negative	Negative interaction pairs (negative examples)
fingerprint	Compound fingerprints

add_model: Used to add model created through machine learning into the DB file

(Format)

cgbvs add_model [option] <db file> <model file> <ID number>

(Description)

Append model file created by SVM machine learning into the DB file while at the same time attaching an ID number to it. The ID number specified here is used for the identification of the negative example set created by the program. Keep in mind that specifying an already used ID number will overwrite an already existing model having the same ID number. By default, it imports the model file that is calculated and created by the SVMlearn command. If the –l option is used, the model file created by the svm-train command of libsvm is imported.

(Option)

-l: Used to import model files created by libsvm

comment: Used to input comments

(Format)

cgbvs comment <db file> <comment> <target>

(Description)

Enter comments regarding what is specified in the <target> argument into the DB file specified in the <db file> argument. Although it is optional, you can enter what you used as compound or protein descriptors. The types of the targets that can be specified are as follows:

compound	Compound descriptors
protein	Protein descriptors
positive	Positive interaction pairs (positive examples)
negative	Negative interaction pairs (negative examples)
fingerprint	Compound fingerprints

create: Used to create an empty DB file

(Format)

cgbvs create [option] <db file>

(Description)

Create a db file with no registered data. If a source file is provided through an option, data such as descriptor information can be imported simultaneously with DB file generation. Even if no option is specified here, the data can be registered by import subcommand later.

(Options)

-c <arg>: Register compound descriptors from the file specified by <arg>.

-p <arg>: Register protein descriptors from the file specified by <arg>.

-i <arg>: Register interaction pairs of the positive examples from the file specified by <arg>.

-n <arg>: Register interaction pairs of the negative examples from the file specified by <arg>.

-f <arg>: Register compound fingerprints from the file specified by <arg>.

The file specified by <arg> should be in CSV format.

delete: Used to remove specific type of data from the DB file

(Format)

cgbvs delete <db file> <target>

(Description)

Deletes the data type specified by the <target> argument from the DB file specified by <db file> argument.

compound	Compound descriptors
protein	Protein descriptors
positive	Positive interaction pairs (positive examples)
negative	Negative interaction pairs (negative examples)
fingerprint	Compound fingerprints

del_model: Used to delete a specified SVM model from the DB file

(Format)

cgbvs del_model <db file> <model ID>

(Description)

Deletes the SVM model having the number specified by <model ID> argument from the DB file specified by <db file> argument. The list of model numbers can be displayed by issuing the "cgbvs status" command. If “all” is specified for the <model ID> argument, all the SVM models will be deleted.

import: Existing data in the db file are deleted before importing new data

(Format)

cgbvs import <db file> <data file> <target>

(Description)

The command imports and registers the data files (CSV), such as descriptor information and interaction pair information into the DB file. The <target> argument specifies the type (descriptor information, interaction pair information, etc. of the compound) of the data file. The types of targets that can be specified are as follows.

compound	Compound descriptors
protein	Protein descriptors
positive	Positive interaction pairs (positive examples)
negative	Negative interaction pairs (negative examples)
fingerprint	Compound fingerprints

The difference with the add subcommand is that it deletes the data type (in the DB file) that is specified in the <target> argument. Use the import subcommand, when you want to register descriptors (such as vector dimensions) that are different from that already registered in the DB file.

(Option)

-m <arg>: Register the contents specified in the <arg> argument as a comment

learn: Used to create input files for machine learning

(Format)

cgbvs learn [option] <db file> <negative example number of sets>

(Description)

Machine learning by SVM is performed after generating the negative example sets using the data (compound descriptors, protein descriptors, the interaction pairs of the positive examples) registered in the DB file (random pair). The model files created are then imported into the DB file. The number of machine learning calculations to be performed by SVM is the same as the number of negative example sets generated. Perform the following procedure when machine learning of negative example sets is to be performed using several machines. First, generate the SVM input files. Once the required number of negative example sets as specified in the <negative example number of sets> argument are generated, perform SVM machine learning for each machine, then import the model files into the DB file.

(Option)

-c <arg>: Specify the C parameter of the soft margin of SVM (default 10)

-g <arg>: Specify the γ parameter of RBF kernel (default 0.01)

-v <arg>: Specify the number of cross-validation iterations (default 5)

-s <arg>: Specify the upper limit of the number of compounds per protein during data sampling

-pc <arg>: Analyze the main components of the compound descriptors and compress the information

-pp <arg>: Perform main component analysis of the protein descriptors and compress the information

When <arg> of the above-mentioned 2 options are integer values, it indicates the number of main components to be sampled. When <arg> is a percentage (numerical %) value, main components are sampled until an accumulative contribution ratio reaches the appointed value.

-m: Generation of negative example sets is not performed

-n: Registered negative example sets will be used

-r: Machine learning is performed without changing a negative example set

When the following two options are specified, only the output of a file is performed, and SVM machine learning is not performed.

-f: The input file to be used for the SVMlearn command is created

-fl: The input file to be used for LIBSVM is created

predict: CGBVS prediction score is performed

(Format)

cgbvs predict [option] <db file> < protein ID> < compound descriptor file>

(Description)

Using the CGBVS model specified by the <db file> argument, the prediction score of the compounds in the file specified by the <compound descriptor file> argument against the target specified by <protein ID> is calculated. Descriptors of the compound to be analyzed are created beforehand and should be in the appropriate file format. There is no upper limit to the number of compounds. Multiple <protein ID> can be specified, separated with commas. “%” can be used as a wild card for a character string, and score is computed for all the proteins registered in the db file by specifying the "all" argument. Available protein targets can be checked by attaching the -p option to the “status” subcommand.

(Option)

-a: Prediction of a target without learned compound information is enabled

-s: Similarity (Tanimoto coefficient) with the known compound group of specified protein is calculated

-d: The value of the decision function of SVM is displayed

-v: Both the binding prediction score and the decision function value are displayed

-n <arg>: A score is computed using only the model ID specified by <arg> argument

status: Information about the model in the DB file is displayed

(Format)

cgbvs status [option] <db file>

(Description)

The information about the model or interaction data registered in the DB file is displayed as a table. When no option is specified, the information about the model is displayed.

(Option)

-c: The compound ID list and the number of proteins which interact are displayed

-p: The protein ID list and the number of compounds which interact are displayed

-pv: All the protein ID lists and the number of compounds which interact displayed

In the case of the -p option, the number of compounds and the protein name can be checked only if the number of compounds is 1 or more. As for the -pv option, all the registered proteins can be checked. The proteins that are listed using the –pv option can be used with the “predict” subcommand.